14351100 B

CONTROL DATA CORPORATION

MP-60

COMPUTER SYSTEM

UPDATE
REFERENCE MANUAL

REVISION RECORD
(Standard Product Manuals)

Revision	Description	Change Authority
9-29-78	Released to Class A	
A	6-15-79 Revised to incorporate ECO 7.7.7.	ECO CB43461
		099
B	See ECO	CB 44951 MC.
(3-5-80)		

Printed in the United States of America
AA 6802 12/78

REVISION LETTERS I, O, Q AND X ARE NOT USED

Use COMMENT SHEET in the back of this manual

LIST OF EFFECTIVE PAGES

New features, as well as changes, deletions, and additions to information in this manual, are indicated by bars in the margins or by a dot near the page number if the entire page is effected. A bar by the page number indicates pagination rather than content has changed.

Page	Rev.	Page	Rev.	Page	Rev.	Page	Rev.
Title Page	B						
ii	B						
iii/iv	B						
v/vi	A						
vii	B						
viii	B						
1-1/1-2	B						
2-1	A						
2-2	B						
2-3	B						
2-4	B						
2-5	A						
↓							
2-8	A						
3-1	A						
↓							
3-3	A						
3-4	B						
3-5							
3-6	B						
↓							
3-12	B						
4-1	-						
4-2	-						
5-1	-						
↓							
5-8	-						
6-1/6-2	-						
7-1	-						
7-2	-						
Comment Sheet	B						

AA 6799 12/78
PRINTED IN U.S.A.
14351100 B

PREFACE

This reference manual describes the system utility program UPDATE used for creating, maintaining, and manipulating a library of programs and data. The MP-60 UPDATE utility is a subset of the CYBER UPDATE utility available on the CONTROL DATA CORPORATION® 6000 and 7000 computer systems.

CONTENTS

CONTENTS (CONT.)

ILLUSTRATIONS

TABLES

INTRODUCTION 1

This reference manual describes the use and operation of the MP-60 UPDATE utility package.

In using UPDATE, the user initially transfers a collection of source decks to a file known as a program library. Each card of each deck is assigned a unique identifier when it is placed on the library, so the card can be referenced during an UPDATE correction run. During correction runs, cards are inserted into or deleted from the program library according to card identification. The card image, even though deleted, is maintained permanently on the program library with its current status (active or inactive) and a chronological history of modifications to the status. A card listed as currently inactive has been deleted and is, in effect, removed from the deck. A card with active status is in the deck; either it has never been deleted or it has been activated after a deletion. During a single UPDATE correction run, a card may undergo one or more modifications, or no modification.

Up to six distinct files are processed in an UPDATE run.

- INPUT Input file containing UPDATE directives and corrective text
- OLDPL Old program library to be updated
- NEWPL New program library to be created by this UPDATE run
- OUTPUT Listable output file
- COMPILE Output file created for input to a compiler or assembler
- SOURCE Output file containing active cards from the program library

All files must use MPX/OS standard blocking format. All files must be opened or equipped before UPDATE execution begins.

INPUT FILE 2

The input file is required and contains UPDATE directives, source decks, and/or corrective text. UPDATE determines the specific operations it is to perform from two sources: parameters on the UPDATE control card, and directives in the input file. In a directive card image, column 1 contains the special character specified as the master control character for this UPDATE run; the default character is *. The control character is followed by a string of characters terminated by a blank or comma. This character string must be recognized by UPDATE as a valid directive.

As each nondirective or data card image is read from an UPDATE input stream, it is assigned a unique card identifier consisting of two parts. The first is an ident, established for the correction set with an *IDENT directive. (Under certain conditions, it may be established by a *DECK or *COMDECK directive.) The second part is a sequence number within that ident. The card identifier, then, is composed of ident.seqnum. An established identifier is retained along with the card image on the old program library (OLDPL). UPDATE directives may reference the card image by its identifier; also, by referencing one ident, groups of card images may be referenced.

Some UPDATE directives exist only in the input stream; other directives are assigned identifiers and are placed on the OLDPL. Since these directives are assigned line identification, they may be referenced thereafter in the same manner as nondirective text card images. Card images exist on the OLDPL in an active or inactive state. For non-directive or data text card images, the state is important only for COMPILE and OUTPUT file output. Only currently active card images are written to the COMPILE file. For UPDATE directives on the library, the state has more meaning. Normally, these directives are processed when the COMPILE file is generated; they are not written to it, but specify certain operations that determine file contents. Inactive directives are ignored.

UPDATE CONTROL CARD

This control card causes the UPDATE program to be loaded from the system library and executed. Parameters on the card specify modes and files for the run. The format is as follows:

UPDATE

```
*UPDATE(p1,p2, . . . ,pn)
```

A right parenthesis terminates the control card. Some parameters are optional and can be in any order. Parameter descriptions are as follows:

C Compile file output

omitted	No compile file output.
C	Compile file output decks are written on the COMPILE file (lun=56); contents are determined by the update mode (F, Q, or normal).
C=lun	UPDATE writes compile file output decks on the named file; contents are determined by update mode (F, Q, or normal).

F Full update

omitted	If the Q parameter is not specified, the omission of F designates the normal (selective) operation mode of UPDATE. All DECKs and COMDECKs are processed. The new program library, if specified, contains all DECKs and COMDECKs after any corrections have been made, and in the sequence in which they occur on the old program library. The source file, if specified, contains all active cards with decks in deck list sequence. The compile file contains all decks corrected during this UPDATE run and all decks specified on *COMPILE directives. Any deck that calls a corrected common deck is also considered to be corrected.
F	Source and compile files, if specified, contain all active decks in old program library sequence. The contents of the new program library are the same as if F were not specified.

I Input

omitted or I	Input is on job INP file (lun=63).
I=lun	Input taken from the named file.

L List output file

omitted or L	List output is written on a file named OUT file (lun=62).
L=lun	List output is written on the named file.

N New program library output

omitted	UPDATE does not generate a new program library.
N=lun	New program library to be written on named file. If lun is not a disk file, UPDATE will write a sequential format PL.

O List options

omitted — List options A and 2 are automatically selected on a creation run. Options A, 2, 3, and 4 are automatically selected on a correction run.

O=$c_1c_2...c_n$ — Each character in the string selects an option. The digit 0 in the string negates any other selections. For example, O=A2340 is equivalent to O=0. The options are as follows:

A Listing of correction set, COMDECK and DECK names.

F All selections other than 0

0 No listing

2 Input directives

Each *IDENT directive produces a page eject. Each card recognized by UPDATE as a valid control card is marked by five asterisks to the left of the card.

3 Commentary on changes to cards processed, consisting of: name of deck, card image, card identifier with sequence number, and a key and action as follows:

Key	Action
A	Inactive card reactivated
D	Active card deactivated
I	Card introduced during run
P	Card was purged during run
Y	Yanked during run

4 Input stream

Cards in error are marked by *ERROR* to the left and to the right. Cards resulting from a *READ directive are marked to the right with the lun of the file from which they were read.

5 Suppress listing of not processed *I, *D, and *R directives. F option will not set this, but O option will clear it.

7 All active cards

9 Listing of all active and inactive cards. List option 3 takes precedence over list option 9. Active (A) or inactive (I) status is indicated to the right of each card image on the listing.

P Old program library

omitted	Must be a creation run or an error message is provided.
P=lun	Old program library written on the named file. If lun is not a disk file, UPDATE verifies that the PL is in sequential format.

Q Quick update (takes precedence over F if F and Q both occur)

omitted	If F is not specified, this is the normal (selective) mode. See F omitted.
Q	Only those decks specified on *COMPILE directives are processed. Corrections that reference cards in decks not specified on *COMPILE cards are not processed. The compile file contains decks specified on *COMPILE directives and any common decks called from the decks. The new program library contains all decks mentioned on *COMPILE directives as well as all common decks encountered on the old program library prior to processing of all of the specified decks. The source file contains the same active cards that are written on the new program library.

S Source output; the contents of the source file are determined by the mode in which UPDATE is operating and the decks named on the *COMPILE directives.

If Q is not selected, regardless of F, the source file contains all cards required to recreate a resequenced library. The recreated library is resequenced because sequencing information is not included on the source file. The source file contains all currently active *DECK, *COMDECK, and *CALL directives in addition to all active text information.

If Q is selected, decks written on the source file are those named on *COMPILE directives and all common decks encountered on the old program library before all explicitly specified decks.

omitted	UPDATE does not generate a source output file unless the source output is specified by the T option.
S=lun	Source output written on the named file.

T Source output excluding common decks

omitted	No source output unless source output specified by S.
T=lun	Source output, excluding common decks, is written on the named file.

* Master control character

omitted	The master control character is * or as redefined during program library creation if this is a correction run
*=character	The master control character, which is the first character of each directive, for this UPDATE run is the character following the equals sign. The character may be A through Z, 0 through 9, + - * / $, or =

On a correction run, if the master control character is not the same as the character used when the old program library was created, UPDATE uses the character indicated on the old program library.

/ Comment control character

omitted	The comment control character is / (slash).
/=character	The comment control character for this UPDATE run is the character following the equals sign. The character may be either A through Z, 0 through 9, + - * / $, or =

UPDATE DIRECTIVES

UPDATE directives are cards in or called into the input stream containing a master control character in column 1, a valid UPDATE directive beginning in column 2, and a blank or comma immediately following the directive. The default master control character * is used in subsequent examples; however, the master control character may be redefined through an UPDATE control card parameter. The master control character is stored in the library information words that prefix the library file created; therefore, the control character obtained from the library will be used. When placed on the library, directives *CALL, *DECK, *COMDECK, *YANKDECK, and *YANK carry with them the master control character with which they were introduced. UPDATE can recognize them on subsequent runs only when the same master control character is given. If file modification is attempted with a master control character other than that specified on OLDPL, a diagnostic will be provided and the program library master control character will be used.

UPDATE directives are written in the following form.

```
*directive
```

or

```
*directive, parameter list
```

* represents the master control character, and the directive follows without intervening spaces. The directive is terminated by a comma or blanks; any number of blanks may appear between the control word and the parameter list. Parameters are separated from each other by commas; embedded blanks are not permitted. Directives may be written in an abbreviated form, as shown in the following table. All numeric values are composed of decimal digits.

Abbreviation	UPDATE Directive Format
*AF	*ADDFILE lun, ident. seqnum
*C	*COMPILE dname1, dname2, . . . , dnamen *COMPILE dnamei. dnamen
*CY	*COPY dname, ident. seqnum *COPY dname, ident1. seqnum1, ident2. seqnum2
*CA	*CALL dname
*CD	*COMDECK dname
*D	*DELETE ident1. seqnum1, ident2. seqnum2 *DELETE ident. seqnum
*DK	*DECK dname
*I	*INSERT ident. seqnum
*ID	*IDENT idname
*P	*PURGE idname1, idname2, . . . , idnamen *PURGE idnamei . idnamen
*PD	*PURDECK dname1, dname2, . . . , dnamen *PURDECK dnamei. dnamen
*R	*RESTORE ident1. seqnum1, ident2. seqnum2 *RESTORE ident. seqnum
*RD	*READ lun
*Y	*YANK idname1, idname2, . . . , idnamen *YANK idnamei. idnamen
*YD	*YANKDECK dname1, dname2, . . . , dnamen *YANKDECK dnamei. dnamen
*/	*/ comments

CARD IDENTIFICATION

UPDATE recognizes one full form and one short form card identifier. The full form is as follows:

ident.seqnum

ident.	1- to 8-character name of a correction set or deck. A period terminates the ident name.
seqnum	Decimal number (1 to 131071) representing the sequence number of the card within the correction set or deck.

Two shortened forms of card identification are provided, which are expanded as follows:

seqnum	Expands to idname.seqnum, where idname is a correction set, DECK, or COMDECK identifier.
.seqnum	Expands to dname.seqnum, where dname is a deck name.

The implied terminology is: ident = idname or dname, dname = DECK or COMDECK name, and deck = DECK or COMDECK.

CYBER UPDATE VERSUS MP-60 UPDATE

Table 2-1 compares the directives available under CYBER and MP-60 versions of UPDATE. Additional restrictions of MP-60 UPDATE compared to CYBER UPDATE are as follows:

1) Control card parameter options are less general. MP-60 UPDATE does not allocate output files SOURCE or NEWPL.

2) The CYBER L and O parameter characters have been reversed on the MP-60 to conform to usage of the L parameter character by other product set members.

3) MP-60 UPDATE processes only disk resident program library files. Tape resident program library requirements must be satisfied through the use of utility programs.

TABLE 2-1. CYBER UPDATE DIRECTIVES VERSUS MP-60 UPDATE DIRECTIVES

CYBER	MP-60
*ABBREV	
*ADDFILE	*ADDFILE
*BEFORE	
*CALL	*CALL
*CHANGE	
*COMDECK	*COMDECK
*COMPILE	*COMPILE
*COPY	*COPY
*CWEOR	
*DECK	*DECK
*DECLARE	
*DEFINE	
*DELETE	*DELETE
*DO	
*DONT	
*END	
*ENDIF	
*ENDTEXT	
*IDENT	*IDENT
*IF	
*INSERT	*INSERT
*LIMIT	
*LIST	
*MOVE	
*NOABBREV	
*NOLIST	
*PULLMOD	
*PURDECK	*PURDECK
*PURGE	*PURGE
*READ	*READ
*RESTORE	*RESTORE
*REWIND	
*SELPURGE	
*SELYANK	
*SEQUENCE	
*SKIP	
*TEXT	
*WEOR	
*YANK	*YANK
*YANKDECK	*YANKDECK
*/	*/
	*FINIS

OLDPL/NEWPL FILE 3

A new program library contains updated decks requested by the user in program library format for use as an old program library in subsequent UPDATE runs. A new program library is originally output by an UPDATE creation run. A program library file contains program library identification data, a deck list, a correction identifier list, and compressed representation of all source cards placed in the library.

Each card image in the program library belongs to a deck. A deck is defined with an active *DECK or *COMDECK directive along with all card images (active or inactive) that follow in the program library up to the next active *DECK or *COMDECK directive. The deck to which a card image belongs is determined by the card position and by the status (active or inactive) of *DECK and *COMDECK directives. Since the *DECK and *COMDECK directives can be deactivated (by *DELETE and *YANK directives), cards belonging to one deck at the beginning of an UPDATE run may belong to a different deck at the end of the run. When a *DECK or *COMDECK directive is deactivated, all card images in that deck become members of the preceding deck.

A source deck is classified as a DECK or a COMDECK, and this classification cannot be changed once incorporated into a program library file. COMDECKs can be called from within other decks as they are being written on the COMPILE file. As the COMPILE file is being generated, each COMDECK call is replaced by the COMDECK active source cards.

Card images are entered into a program library during the library creation step or are inserted from a correction set during program library update steps. Each card image is assigned a unique identifier (the *DECK, *COMDECK, or *IDENT identifier) and a sequence number relative to that identifier as it is added to the program library. Groups of card images can be removed from the program library by the *PURGE and *PURDECK directives.

Cards are assigned the active status when they are added to the program library. The card status is changed to the inactive status by the *DELETE directive. The status of groups of card images may also be altered through the *YANK and *YANKDECK directives.

*YANK, *YANKDECK, *PURGE, and *PURDECK can be used to recreate a predecessor program library.

LIBRARY FILE CREATION

UPDATE operates in creation or correction mode. UPDATE considers any run to be a creation run if the first directive it encounters in the input file (except for */, *COMPILE, or *READ) is a *DECK or *COMDECK directive. Even if an old program library file is defined to the task, UPDATE will ignore its existence and process the run in creation mode.

In a creation run, each *DECK or *COMDECK directive in the input stream defines a deck to be incorporated into the library file. The directives also cause entries to be made in the ident and deck lists. UPDATE identifies and sequentially numbers each card image written to the library file. Identifiers take the form, dname.seqnum; dname is the name associated with the deck, and seqnum is the sequence number. Numbers are assigned, starting with 1 for the *DECK or *COMDECK card, and including all input file text cards and those introduced by a *READ directive.

DECK

*DECK dname *DK	dname	1- to 8-character deck name for this deck that is different from any other deck name in the deck list; legal characters are A through Z, 0 through 9, and + - * / $ =. The first character must be A through Z.

COMDECK

*COMDECK dname *CD	dname	1- to 8-character name of deck being introduced; this name must differ from any names already in the deck list. Legal characters are A through Z, 0 through 9, and + - * / $ =. The first character must be A through Z.

The primary difference between *DECK and *COMDECK is that the latter introduces a deck that may be called from other decks as they are being written on the compile file; however, common decks must precede other decks that may call it.

When text and directives are to be read from files other than the input stream, file manipulation directives are used. The directives may appear anywhere in the input stream; they may appear only on the main input file and may not reference files specified for internal use by UPDATE, such as those identified by file parameters. To read from a file other than the input stream, the user program should include the control statement.

READ

*READ lun *RD	lun	Logical unit number of file containing insertion text and/or directives; *READ and some forms of *ADDFILE are illegal in the file, lun.

The *READ directive causes UPDATE to temporarily stop reading the input file and begin reading directives and inserting text from the named file at its current position. UPDATE reads from this alternate directives file until it encounters an end-of-file mark and then resumes reading the next card from the primary input file. An example is as follows:

```
MORE TEXT
*READ 10                    INSERTION TEXT IS READ FROM
                            LUN = 10
*COMDECK SAMPLE
```

LIBRARY CORRECTION DIRECTIVES

The library correction process is the most common use of UPDATE. UPDATE directives and data card images comprise the correction set. The directives provide UPDATE with instructions concerning the modification of the program library text stream. The old library is not destroyed; the corrections are permanent only in the new program library being created.

A correction run consists of a read and process input stream phase and a correction phase. During the first phase, UPDATE reads directives and text, adds new decks, and constructs a dictionary of requested correction operations. During the second phase, UPDATE performs the requested modifications on a deck-by-deck basis.

Modifications for each correction set are performed by UPDATE in the order in which sets are introduced. The order is irrelevant if no correction is dependent on another. However, if a dependent relationship exists, order is of paramount importance.

Correction directives cause card images to be inserted, deactivated, or reactivated in program library decks according to card sequence number. A card is identified by deck or ident name and sequence number. Each new card is assigned a correction set identifier specified by the user. UPDATE sequences the new cards. All cards having the same

correction identifier comprise a correction set. UPDATE permits a user to remove (yank) the effects of a correction set (or deck) and later restore the set (or deck). This feature is convenient for testing new code. Requests for yanking are maintained in the YANK$$$ deck. Before obeying a correction, UPDATE checks the correction identifier against the YANK$$$ deck to see if the correction has been yanked.

UPDATE also allows a complete and irreversible purging of correction sets and decks (through the *PURDECK and *PURGE directives).

The following directives are used for correcting and updating a program library.

ADDFILE

*ADDFILE lun, ident. seqnum

lun	Logical unit number of the file from which information is to be read. The text cannot contain correction directives. If lun is omitted, it is assumed to be the UPDATE input file (lun = 63 or its equivalent, as specified by the I parameter).
ident. seqnum	Identifier and sequence number of card after which decks are to be placed on program library. If ident. seqnum is omitted, the addition will occur at the end of the program library. If this parameter is present, the full form must be used.

When *ADDFILE is encountered, UPDATE reads creation directives and text data from the named file (lun) and inserts this information after the specified card (ident. seqnum) of the program library. The first card of lun must be *DECK or *COMDECK. UPDATE reads from this file until an end-of-file, which returns UPDATE to the main input file. If the lun on *ADDFILE is the main input file, cards are read until the next UPDATE directive. UPDATE directives placed on the library (*DECK, *COMDECK, *CALL) do not terminate the *ADDFILE function. *ADDFILE directives with lun not equal to the alternate input file lun are illegal on an alternate input file. A *READ directive is illegal during processing of the add-file unless the add-file is the main input file.

Either or both of the lun and ident. seqnum parameters may be omitted.

CALL

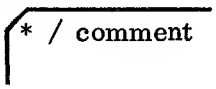
*CALL dname
*CA

dname	Name of common deck to be written on compile file.

UPDATE writes the text of a previously encountered common deck (dname) onto the COMPILE file. Common code may be declared in a common deck and used in subsequent decks without repeating the cards.

The *CALL card does not appear on the COMPILE file. The contents of the common deck, excluding the *COMDECK card, replace the *CALL card. Common decks can call other common decks. However, to avoid circularity of calls, a common deck must not call itself or call decks that contain calls to the common deck. A *CALL directive is effective only when it is within a deck either directly or as a result of replacing a *CALL with the COMDECK text.

COMMENT



Comments to be listed with a correction set may be included with a correction set with a comment directive. It has a master control character (*) in column 1 and a comment control character (/) in column 2. This card is ignored by UPDATE except that it is copied onto the listable output file. A comment may appear at any place in the input stream.

PROGRAM LIBRARY FORMAT

The program library identification data records the UPDATE version used to create the library file, the library master control character, and the date and time of library creation.

The deck list records the DECK and COMDECK (deck) identifiers and the library file block address for the start of each deck in the library. This list is recorded in the order of deck occurrence in the library.

The card identifier name (idname) list records the identifier of every DECK, COMDECK, and correction set represented in the library. This list is recorded in the order in which the identifiers were introduced into the library.

UPDATE program libraries have the format shown in Figures 3-1 and 3-2.

In 'RAN' only, the program library, the character string 'UPDATE', and the version number are used to verify an OLDPL at the beginning of the update process. The program library master control character is required by UPDATE to recognize control cards as they are encountered during the processing of user decks being written to the COMPILE file. The date and time of program library creation are recorded as a user maintenance aid. The block numbers record the first and 'last+1' blocks of the ident and deck name lists.

Each entry in the dname list has the following format.

word 1	dname(1)	dname(2)	dname(3)	dname(4)
word 2	dname(5)	dname(6)	dname(7)	dname(8)
word 3	type / flags		blth	

dname — One to eight alphanumeric characters representing the deck name. The name is left adjusted in the 8-character field and blank filled to eight characters.

type — Type of identifier represented.

Entry	Indicates
010	DECK identifier
100	COMDECK identifier

flags — UPDATE execution-time flags.

blth — First block number of the deck.

The first deck name is always YANK$$$. The dname list is ordered according to the arrangement of decks in the program library.

Each entry in the idname list has the following format.

word 1	ident(1)	ident(2)	ident(3)	ident(4)
word 2	ident(5)	ident(6)	ident(7)	ident(8)
word 3	type / flags		(SN)	

ident — One to eight alphanumeric characters representing a card identifier. The name is left adjusted in the 8-character field and blank filled to eight characters.

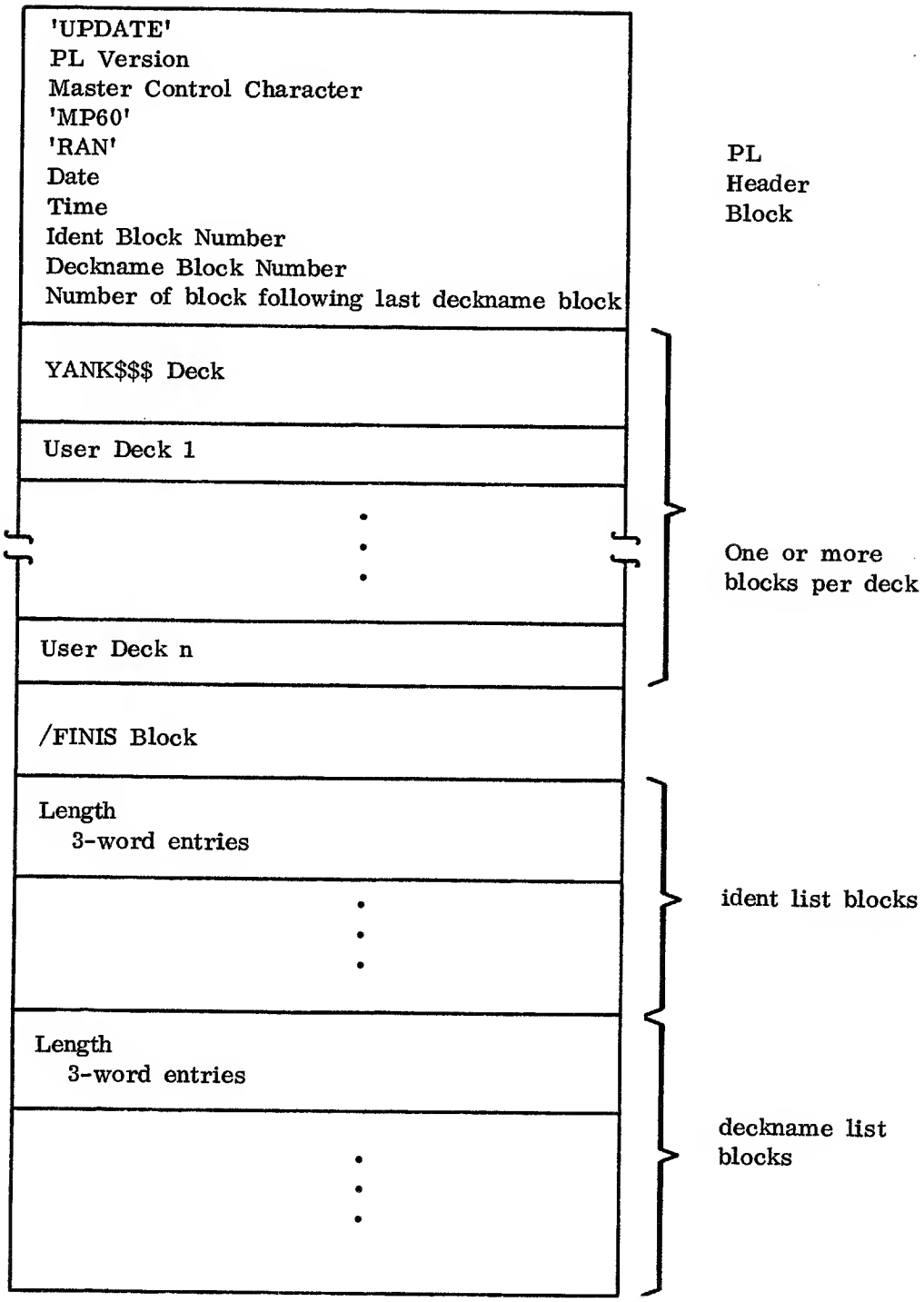


Figure 3-1. Random PL Format

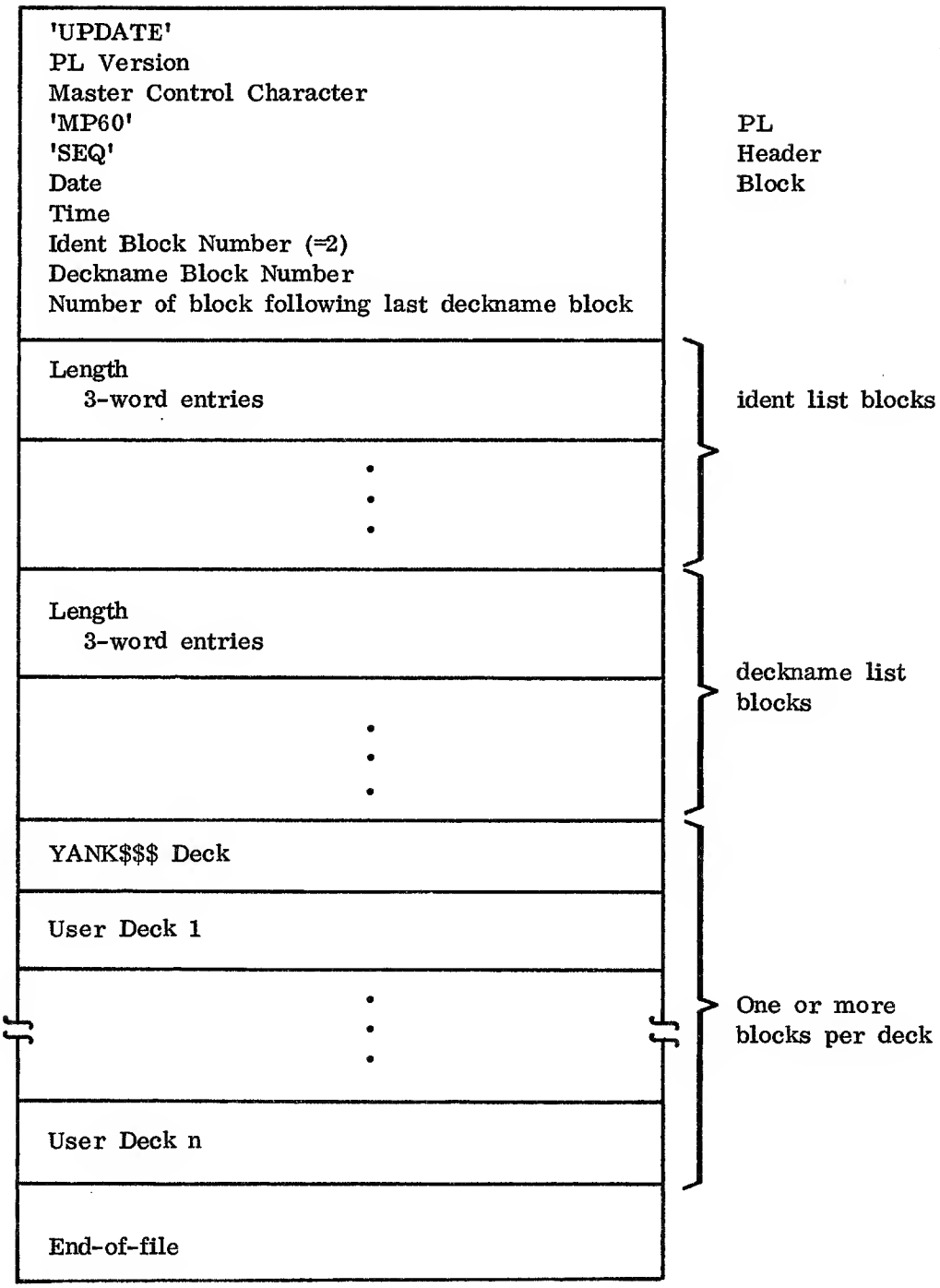


Figure 3-2. Sequential PL Format

type — Type of identifier represented.

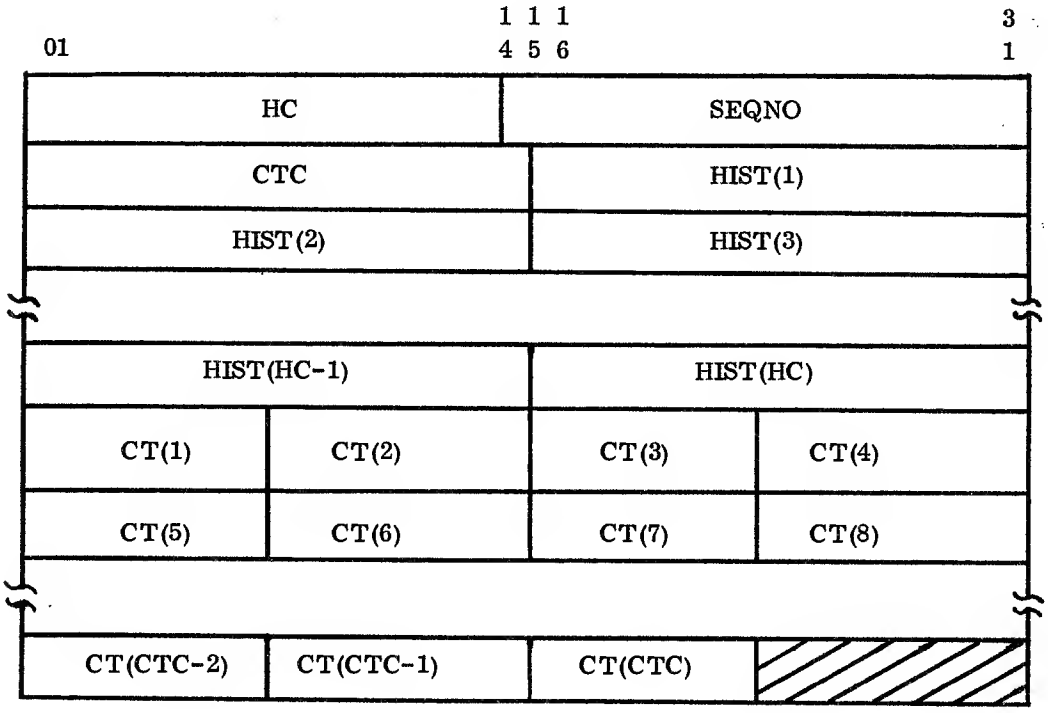
Entry	Indicates
010	DECK identifier
100	COMDECK identifier
001	IDENT identifier

flags — UPDATE execution-time flags.

SN — Records sequence number assignment during UPDATE correction run execution.

The ident list is ordered according to the sequence of introduction of the idents into the program library. The position of the ident in the list is used in the history bytes accumulated for each card. Idents which are purged from the library are replaced by all blanks in the ident list; this preserves the ordering of idents and allows reuse of purged idents.

Each user deck in the program library begins on a new block boundary. Each card image within each deck begins on a word boundary. The number of words allocated to each card image is variable, including a correction history and a compressed representation of the card. The format of each card image is as follows:

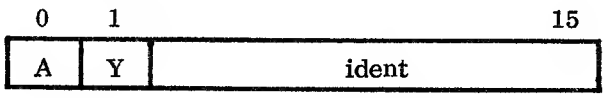


HC History parcel count (15-bit field)

SEQNO Card sequence number (relative to HIST(1), 17-bit field)

CTC Compressed text byte count (16-bit field)

HIST Correction history byte (16-bit field); structured as follows:



A — Activation status bit

Entry	Indicates
0	Card not active
1	Card active

Y — Yank status bit

Entry	Indicates
0	Card not yanked
1	Card yanked

ident — Number of entry in ident list which contains the card identifier character string

CT — Compressed text bytes. Each byte is the original ASCII character except that consecutive blanks are replaced by a 2-byte sequence, the ASCII null character (00) followed by the blank count.

PROGRAM LIBRARY LIFE CYCLES

During development, release, and maintenance of a software product, UPDATE provides the means to have a controlled process for making additions, deletions, and changes to that product. To ensure that all organizations working on a software product are aware of all changes made, an interim program library (basically, a NEWPL not released that consists of a released OLDPL and changes) will be available when requested. To provide an effective software product maintenance history, the following discipline should apply.

1) The released program library contains a complete set of binary modules on a companion file.

2) Corrections to the software product results in the generation of a COMPILE file, either through use of the Q mode and *COMPILE directives or through a normal-mode UPDATE run.

3) The COPYL utility then can be used to combine the release equivalent binaries with the binaries obtained in step 2. The resultant binary can then be tested.

4) The accumulated corrections can then be used as input to an UPDATE run for creation of an interim program library. An interim set of binary modules must also be created to allow for testing.

5) Corrections can now be developed against the interim program library created in step 4.

6) Steps 4 and 5 can be repeated as often as desired to facilitate the development/ maintenance activities. The interim program library is always constructed from the completed corrective code and the last released program library (OLDPL).

The interim program library provides efficient use of UPDATE where dependencies exist in corrections (one correction affects another line of corrective coding) because of the method of assigning line identifiers. Without an interim program library, more than a minimum number of modules must be compiled/assembled on each test run to ensure correct assignment of line identifiers and to avoid additional diagnostics.

The software product may undergo several release cycles, and, at some point, it may be desirable to reconstruct the program library. Reconstruction reduces the amount of file space used by the program library by purging all inactive lines. Reconstruction is accomplished by requesting the source file output and using the source file as an UPDATE input file. The reconstruction process incorporates all corrective code into the created source file and discards the maintenance history and all YANK and YANKDECK control information. Because the card identifiers are redefined, all development/maintenance activity must be suspended until the new card identification information is available.

OUTPUT FILE 4

The content of the OUTPUT file varies, depending on the nature of the run and the output options selected on the UPDATE control card. For creation runs, UPDATE provides a list of known *DECKS, *COMDECKS, all error messages, and all active control cards. For correction runs, the preceding are augmented to include all cards for which status changed during this UPDATE run and all cards encountered in the input file.

LISTING FORMATS

There are three listing formats associated with OUTPUT. These consist of input stream directives, input stream new text, and correction commentary.

INPUT STREAM DIRECTIVES

```
*****     *CA  COMCOB                                        CU1C
*****     *D  COB30.1130
                IF(CRC(1, TXT(ITP1, 3)).EQ. 0)GOTO 40
*****     *D  COB30.1173, COB30.1176
*****     *I  COB30.1177
*****     *CA  COMCOB                                        CU1C
*****     *D  COB30.1199, COB30.1202
*****     *CA  COMCOB                                        CU1C
*****     *D  COB30.1258, COB30.1261
*****     *I  COB30.1262

*ERROR*     *****     ATTEMPT TO CALL NONEXISTENT DECK     *****     *ERROR*
*ERROR*     *****     ATTEMPT TO CALL NONEXISTENT DECK     *****     *ERROR*
```

INPUT STREAM NEW TEXT

```
*****     /C SSI
*****     /D STI1.111
               OP3=OP3.AND..NOT.15                         MODS
*****     /C STI1
*****     /D TER.106, 108
          30    CONTINUE                                    MODS
*****     /C TER
*****     /AF 63, REA2.21
```

```
*****    /DK READMS      XFER DATA FROM MEM TO DISK BLOCK      READMS
               SUBROUTINE READMS(LUN, ARY, LTH, BLK)          READMS
               INTEGER ARY(1), BLK                            READMS
               CALL FTNULOC(59, BLK)                          READMS
               BUFFER IN(59, 1)(ARY(1), ARY(LTH))             READMS
               II=IFUNIT(59)                                  READMS
```

CORRECTION COMMENTARY

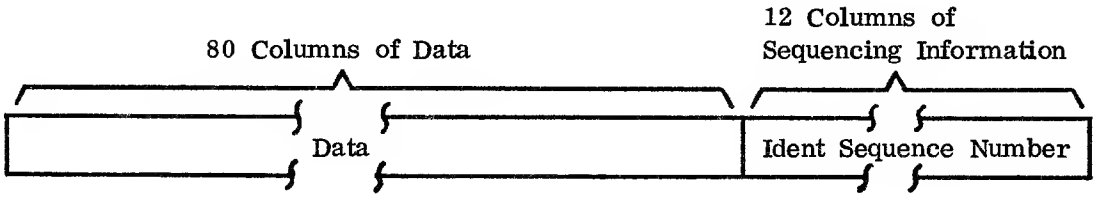
```
COB40     C                                                  COB40   504 D
COB40     C     SET SECTOR WHERE *T WRITTEN FOR LOADER.      COB40   505 D
COB40     C                                                  COB40   506 D
COB40           IF(LO(6).EQ.0) GO TO 24                      COB40   507 D
COB40           I1=OBP1-1                                    COB40   508 D
COB40           ASSEM $C800,I1,   $60E4                      COB40   509 D
COB40           GO TO 26                                     COB40   510 D
COB40     24    OBJ(1)=ASC(2R*T)                             COB40   511 D
COB40           OBJ(2)=3328                                  COB40   512 D
COB40           CALL OPS(6,-1)                               COB40   513 D
COB40     26    CONTINUE                                     COB40   514 D
```

COMPILE FILE 5

The compile file is a primary form of output from UPDATE. It contains updated source card images to be assembled, compiled, or used as program data. The compile file output is optional; it is selected for output through the C option on the UPDATE control card.

The decks written to the COMPILE file are selected, in part, by the F and Q UPDATE control card parameters. Through the use of F and Q, the normal, full, or quick update modes can be selected. During a normal UPDATE run (F and Q not selected on the UPDATE control card), UPDATE writes on the compile file all decks specified on *COMPILE directives as well as all decks corrected during the run. During a full UPDATE run (F selected and Q not selected on the UPDATE control card), all decks are written on the COMPILE file. During a quick UPDATE run (Q selected on the UPDATE control card), only decks specified on *COMPILE directives are written on the compile file. Decks are always written in the order of occurrence in the program library.

The compile file format for each card consists of 80 columns of data followed by 12 columns of sequence information; a card image is 92 columns. The expanded card image is as follows:



UPDATE attempts to place the maximum sequence information in the 12 columns allocated. The 1- to 8-character ident is left adjusted in column 81 and the sequence number is right adjusted in column 92. If the ident and sequence number exceed the 12 column field, UPDATE truncates the least significant characters of the ident, leaving the sequence number intact.

In the following example, ident is SEVENCH and the sequence number is 1144 or 114400.

Column	81								89			92
	S	E	V	E	N	C	H		1	1	4	4
	S	E	V	E	N	C	1	1	4	4	0	0

COMPILE DIRECTIVES

COMPILE

The *COMPILE directive has two formats as follows:

```
*COMPILE dname1,dname2,...,dnamen
*C
```

```
*COMPILE dnamei.dnamen
*C
```

dname	Name of deck to be written on the compile file, new program library, and source file.

The first form of the directive requests one or more decks that may be in any sequence on the library. The second form requests all decks in the deck list starting with dnamei through dnamen. If UPDATE fails to find the named deck or if the deck names are reversed, a diagnostic message is provided.

*COMPILE directives can occur anywhere in the input stream.

COPY

The COPY directive has two formats as follows:

```
*COPY     dname, ident. seqnum
*CY
```

```
*COPY     dname, ident1. seqnum1, ident2. seqnum2
*CY
```

dname	Deck on old program library that contains the cards to be copied.
ident. seqnum	Card identifier for single card to be copied.
ident1. seqnum1, ident2. seqnum2	Card identifiers of first and last cards in sequence of cards to be copied. ident1. seqnum1 must occur before ident2. seqnum2 on the library. The range can include cards already copied which are affected by the *COPY.

The COPY directive can occur where input text may be defined in the input file; that is, the COPY directive can follow an *I, *D, or *R correction directive, or it may occur in text which follows an *I, *D, or *R directive.

The COPY directive results in suspension of input from the primary or alternate input file. During the suspension, the old program library is accessed to locate and read the designated card or cards. All previous sequence information is lost; the new lines are treated as if they had been read from the input file. When the identified cards have been read or the end of the deck is encountered, reading of the primary or alternate input file is resumed.

DELETE

The *DELETE directive has two formats as follows:

```
*DELETE ident.seqnum
*D
```

```
*DELETE ident1.seqnum1, ident2.seqnum2
*D
```

ident.seqnum	Card identifier for single card to be deleted.
ident1.seqnum1, ident2.seqnum2	Card identifiers of first and last cards in sequence of cards to be deleted. ident1.seqnum must occur before ident2.seqnum on the library. The range can include cards already deleted which are affected by the *DELETE.

With the *DELETE directive, the user deactivates a card or block of cards and, optionally, replaces it with insertion cards which follow the *DELETE directive.

A deactivated card remains on the library and retains its sequencing. It can be referred to in the same way as an active card. A deactivated card is not included in compile file or source file output.

IDENT

The *IDENT directive has the following format.

```
*IDENT idname
*ID
```

idname	1- to 8-character identifier to be assigned to this correction set. Legal characters are A through Z, 0 through 9, and + - * / $ =. The first character must be A through Z.

This name causes a new entry in the ident list. Each card inserted by the correction set and each card for which the status is changed receives a correction history byte that indexes this idname. Sequencing of new cards begins with one for this idname. Omitting idname causes a format error. If idname duplicates a name previously used, UPDATE issues an error message. Both errors are nonfatal.

This idname remains in effect until UPDATE encounters another *IDENT directive or a *PURGE, *PURDECK, or *ADDFILE directive.

INSERT

The *INSERT directive has the following format.

```
*INSERT ident.seqnum
*I
```

ident	Identifier name.
seqnum	Decimal sequence number.

Cards may be inserted in a deck with the *INSERT directive. The ident and seqnum specify the card after which the insertion is to be made.

PURDECK

The *PURDECK directive has two basic formats as follows:

```
*PURDECK dname1, dname2, ..., dname_n
*PD
```

dname	Deck names for decks to be purged.

```
*PURDECK dname_i.dname_n
*PD
```

A *PURDECK directive causes the permanent and nonreversible removal of a deck or group of decks from the program library. Every card in a deck is purged, regardless of the ident to which it belongs. *PURDECK does not purge idnames. Thus, a deck name purged as a result of *PURDECK can be reused as a dname in a later UPDATE run.

A *PURDECK directive can be any place in the directives input. It terminates any previous correction set. Therefore, *INSERT, *DELETE, or *RESTORE cannot follow a *PURDECK directive, but must come after an *IDENT directive. The YANK$$$ deck cannot be purged. Purging cannot be rescinded.

The deck named $dname_i$ and all decks up to and including $dname_n$ listed in the deck list are purged. If $dname_i$ and $dname_n$ cannot be located or are in reverse order, UPDATE issues an error message.

PURGE

The *PURGE directive causes the permanent and nonreversible removal of a correction set or group of correction sets. A *PURGE directive can appear anywhere in the directives input. Because it terminates a previous correction set, an *INSERT, *DELETE, or *RESTORE cannot follow a *PURGE directive; they must follow an *IDENT directive. Purging cannot be rescinded.

The *PURGE directive has the following two basic formats.

*PURGE idname1, idname2, . . . , idname$_n$
*P

idname	Identifiers for correction sets to be purged.

*PURGE idname$_i$.idname$_n$
*P

Correction set idname$_i$ and all sets up to and including idname$_n$ on the directory are purged. If idname$_i$ and idname$_n$ cannot be located or are in reverse order, UPDATE issues an error message.

If UPDATE cannot locate a specified correction set, it issues an error message. Purged idnames can be reused on subsequent correction sets in later UPDATE runs, providing they do not appear in the YANK$$$ deck.

YANK

The *YANK directive has the following two basic formats.

*YANK idname1, idname2, . . . , idname$_n$
*Y

idname	Name of correction set previously applied to the program library. If UPDATE fails to find idname, it issues an error message.

*YANK idname$_i$.idname$_n$
*Y

The correction set idname$_i$ and sets up to and including idname$_n$ on the directory are yanked. If idname$_i$ and idname$_n$ cannot be located or are in reverse order, UPDATE issues an error message.

UPDATE places the *YANK directive in the YANK$$$ deck. During the modification phase, UPDATE checks each correction to see if it has been yanked. All yanked corrections are ignored. If the card was deactivated by the yanked correction set, UPDATE reactivates it. If the card was activated by the yanked correction set, UPDATE deactivates it. Thus, UPDATE changes the correction history byte for every card that has changed status.

A *YANK must be part of a correction set.

A *YANK directive does not terminate insertion.

Example:

```
*YANK OLDMOD
```

This directive causes all effects of the correction set OLDMOD on the entire library to be nullified. Cards introduced by OLDMOD are deactivated; cards deactivated by OLDMOD are reactivated; cards reactivated by OLDMOD are redeactivated.

YANKDECK

The *YANKDECK directive deactivates all cards within the decks specified. The directive format is as follows:

```
*YANKDECK dname1,dname2,...,dname_n
*YD
```

dname	Name of deck to be deactivated. All cards in the deck are deactivated, regardless of the correction set to which they belong. If UPDATE is unable to find dname, it issues an error message.

```
*YANKDECK idname_i.dname_n
*YD
```

The correction deck $dname_i$ and all decks up to and including $dname_n$ on the directory are yanked. If $dname_i$ and $dname_n$ cannot be located or are in reverse order, UPDATE issues an error message.

The YANK$$$ deck cannot be deactivated by the yank operation.

The *YANKDECK directive must be part of a correction set.

*YANKDECK does not terminate insertion.

Example:

```
*YANKDECK OLDDECK
```

This directive affects all cards in OLDDECK, regardless of the correction set to which they belong.

RESTORE

The *RESTORE directive has the following two formats.

```
*RESTORE ident.seqnum
*R
```

```
*RESTORE ident1.seqnum1, ident2.seqnum2
*R
```

ident.seqnum	Card identifier for single card to be restored.
ident1.seqnum1, ident2.seqnum2	Card identifiers of first and last cards in sequence of cards to be restored. ident1.seqnum must occur before ident2.seqnum on the library.

With the *RESTORE or *R directive, the user reactivates a card or block of cards. New text may be inserted after the last reactivated card.

ERROR CONDITIONS

UPDATE can detect four overlapping correction situations.

Type 1	Two or more modifications are made to one card by a single correction set.
Type 2	A modification attempts to activate an already active card.
Type 3	A modification attempts to deactivate an already inactive card.
Type 4	A card is inserted after a card which was inactive on the OLDPL.

When any of these types is detected, UPDATE prints the offending line with the words TPn OVLP appended on the far right. Detection of an overlap does not necessarily indicate a user error. Overlap messages are advisory and point to conditions in which the probability of error is greater than normal.

Type TP2 and TP3 are detected by comparing existing correction history bytes with those to be added. Complex operations involving *YANK and *PURGE may generate these overlap messages even though no overlap occurs.

SOURCE FILE 6

The source file contains a copy of all active *DECK and *COMDECK directives and all active cards within each deck. The source file is an optional output from UPDATE selected through the use of the S and T options on the UPDATE control card. Once created, the file can be used as source input on subsequent UPDATE runs. The source file contains 80-column card images.

When Q is not selected on the UPDATE control card, the source file generated contains all cards needed to create a program library. The source file can be used as input for an UPDATE run that produces a resequenced program library with all the inactive cards purged.

When Q is selected on the UPDATE control card, the source file contains all decks requested on the *COMPILE directives and any common decks encountered prior to processing all of the specified decks. The only directives that can legally appear in the source file are *DECK, *COMDECK, and *CALL.

SAMPLE DECK STRUCTURES 7

CREATION RUN DECK STRUCTURE

To create a NEWPL from an ASCII source deck, the input is from the standard input unit, INP. UPDATE output is to a file named 'UPDATE'.

```
*JOB(...)
*SCHED(...)
*OPEN(11, UPDATE, NPL, 1, ...)
*UPDATE(N=11, ...)
/DK     'DECK AND/OR COMDECK NAME'
        (ASCII SOURCE DECK)
/FINIS
*EOJ
```

CORRECTION RUN DECK STRUCTURE

To generate a NEWPL by correcting an OLDPL from an ASCII source deck, input is from the standard input unit, INP. UPDATE output is to a file named 'UPDATE', edition 2.

```
*JOB(...)
*SCHED(...)
*OPEN(4, UPDATE, NPL, 1, ...)
*OPEN(11, UPDATE, NPL, 2, ...)
*UPDATE(P=4, N=11, ...)
/ID     'CORRECTION SET IDENTIFIER'
        (ASCII CORRECTION SOURCE DECK)
/FINIS
*EOJ
```

COMPILE RUN DECK STRUCTURE

To produce a compile file from a program library, names of decks are processed from the *COMPILE directive card and UPDATE produces a file named 'UPDATE', 'CFIL'.

```
*JOB(...)
*SCHED(...)
*OPEN(11, UPDATE, NPL, 2, ...)
*OPEN(2, UPDATE, CFIL, 1, ...)
*UPDATE(P=11, C=2, ...)
/C   'NAMES OF DECKS'
/FINIS
*EOJ
```

COMMENT SHEET

MANUAL TITLE MP-60 Computer System UPDATE Reference Manual

PUBLICATION NO. 14351100 REVISION B

FROM: NAME:

BUSINESS ADDRESS:

COMMENTS:

This form is not intended to be used as an order blank. Your evaluation of this manual will be welcomed by Control Data Corporation. Any errors, suggested additions or deletions, or general comments may be made below. Please include page number references and fill in publication revision level as shown by the last entry on the Record of Revision page at the front of the manual. Customer engineers are urged to use the TAR.

CUT ALONG LINE

AA3419 REV. 11/69 PRINTED IN U.S.A.

NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

FOLD ON DOTTED LINES AND STAPLE

STAPLE

STAPLE

FOLD

FOLD

FIRST CLASS
PERMIT NO. 8241

MINNEAPOLIS, MINN.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION
Aerospace Division
Box 609, HQG346B
Minneapolis, Minnesota 55440

CUT ALONG LINE

FOLD

FOLD